On Convolutional Lattice Codes and Lattice Decoding using Trellis Structure

Mehdi M. Molu, Kanapathippillai Cumanan, Manijeh Bashar and Alister Burr

***Abstract*—Constructing hypercubic lattices from convolutional codes based on Construction A and D is investigated in this paper and their error performance in a point to point communication system is studied. Moreover, analogous to Construction A/D, single/multilayer Code Lattices are proposed. As Construction D requires certain minimum Euclidean distance criteria, we propose methods to guarantee the distance requirements of Construction D which results in a superior lattice construction compared to Construction A. Due to the key role of soft input soft out decoding algorithms in improving the performance of a code, lattice decoding based on the BCJR algorithm for lattices constructed from convolutional codes is also proposed in this paper. Moreover, as the BCJR algorithm requires knowledge of the statistical characteristics of modulo lattice additive noise (MLAN), the probability density function of MLAN is derived in closed form.**

***Index Terms*—Lattice Encoding/Decoding, Convolutional Lattices, BCJR Lattice Decoder, Construction A/D, Modulo-Lattice Additive Noise**

I. Introduction

Constructing lattices from Forward Error Correction (FEC) codes has been a rather active field of research in the past, and has led to, e.g., Constructions A, B, C, D and etc. [1], [2]. In this paper we are also interested in lattice construction, and in particular, Construction A because of its simplicity and Construction D because of the potential performance of the resulting codes. For lattice construction, we use convolutional codes as the underlying FEC code because capacity approaching Turbo codes consist of two (or more) convolutional codes; therefore, constructing convolutional lattices is a major step forward towards constructing Turbo lattices. There has been, surprisingly, little work on exploitation of convolutional codes for constructing lattices reported in the literature. Although [3], [4] discusses lattices based on convolutional codes (indeed, Turbo codes), the transmitted signals are restricted to be binary which loses the freedom to arbitrarily specify the rate of the lattice code: in this paper we extend this to allow non-binary transmission with arbitrary transmission rate; moreover, lattice decoding algorithms have not been discussed in [3], [4], whereas we propose adopting the trellis structure of the underlying convolutional code for lattice decoding and demonstrate superior performance using this approach. This provides the possibility of implementing computationally feasible lattice decoding methods for convolutional lattices. Note that universal lattice decoding methods commonly applied in the literature for practical communication systems (e.g. sphere decoding) have until now been relatively complex, and as a result are applicable only to lattices with very short dimension [5]–[8]; for instance, a lattice decoder was proposed in [7] with relatively reasonable complexity that was examined for lattices of dimension up to 32. Indeed this is a major drawback because the code length (lattice dimension) of real communication systems is much longer than 32. Consequently, the lattice decoding method proposed in this paper for decoding convolutional lattices is practically important due to its feasible complexity at high dimensions; furthermore, it will be observed in Sec IV-C that the proposed lattice decoding approach significantly outperforms existing lattice decoding algorithms. [9] also studies convolutional lattices, however, the proposed scheme is mostly "attractive for Inter Symbol Interference (ISI) channels". Decoding algorithms of other ISI channel, in particular, Faster Than Nyquist (FTN) signalling has been studied by the authors in [10], however note that [9] considers code filters combined with ISI filters which results in unification of equalization and decoding. In this paper we are not interested in ISI channels nor FTN signalling but we would like to construct lattices from convolutional codes that are proved to approach capacity when applied in Turbo codes. Moreover, [9] considers single layer lattices whereas we assume multilayer as well as single layer lattices.

An earlier version of this work was reported in [11], however, due to the superior performance of Construction D over Construction A, [11] is further extended in this paper and construction of convolutional lattices based on Construction D and multi layer Code Lattices are also investigated in this paper. Construction D relies on two characteristics of the underlying FEC codes: (i) the codes are nested as a chain of sub-codes and (ii) these sub-codes have larger Minimum Euclidean Distance (MED) than the parent code [1, Ch. 8]. Convolutional codes do not readily fulfil such requirements, which may be one reason that convolutional codes have not so far been exploited in Construction D[1]. In this paper, we first propose constructing convolutional lattices based on Construction D by neglecting the MED criterion, and then introduce means of increasing the MED of nested convolutional codes by rearranging the input messages which guarantees to fulfil the MED criterion of the Construction D definition.

For the convolutional lattices based on Construction A/D (and also single/multi layer Code Lattices),

This work was supported in part by the European Commission FP7 contract no. 318177 (DIWINE), and in part by the EPSRC under Grant EP/K040006 (NetCoM5G). M. M. Molu is with the 5G innovation Centre (5GIC) at the University of Surrey, UK. Email: m.molu@surrey.ac.uk. K. Cumanan, M. Bashar and A. Burr are with the department of Electronics, University of York, UK. Email: {kanapathippillai.cumanan,mb1465,alister.burr}york.ac.uk

[1]Although [3], [4] study Turbo lattices based on Construction D, they neglect the minimum distance criterion, and consequently it may result in degradation of the lattice code performance.

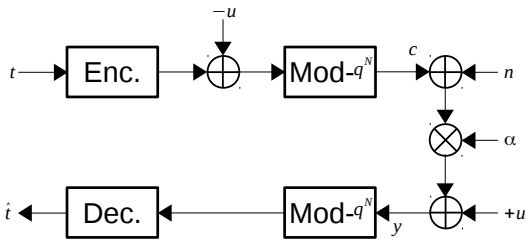

Fig. 1. System Model

equivalent encoding based on shift registers is proposed enabling us to exploit existing decoding algorithms of convolutional codes for convolutional lattices too. The lattice codes based on convolutional lattices allow optimal lattice decoding using the trellis structure of the underlying convolutional code; e.g., the BCJR algorithm. A further contribution of the paper is to provide methods to incorporate the BCJR algorithm in lattice decoding. This requires the statistical characteristics of Modulo Lattice Additive Noise (MLAN), and therefore we also derive the probability density function (pdf) of MLAN in closed form for lattices with hypercubic shaping regions. A rather similar pdf has been described in [12, Sec. III-B], however, no closed-form expression for the pdf was derived.

The new lattice decoding algorithms we develop are based on ML/MAP decoders, and thus have similar complexity. However, throughout the paper it will be observed that on the point to point channel ML/MAP decoders outperform the corresponding lattice decoders in practice (i.e., in dimensions less than infinity [13]–[16]). This might raise the question of the benefits of lattice decoding as compared to pure ML/MAP decoding. Our motivation, however, extends beyond the point-to-point channel to relay communication systems [17], [18], and in particular, recently-proposed communication paradigms such as Compute and Forward (C&F) [19], which relies purely on the structure of the lattice, and hence requires practical implementation of lattice decoding for lattices with arbitrarily high dimension, for which direct ML/MAP decoding would be prohibitively complex. Here we study lattice encoding and decoding algorithms in a point to point communication system as a step towards their use in C&F decoders.

This paper is organized as follows: In Section II a point-to-point system is introduced. In Sec III the statistical characteristics of MLAN are studied and in Sec IV convolutional lattices based on Construction A are proposed, along with the methods for lattice decoding. An example is presented in section V that exploits the proposed lattice encoding and decoding algorithms in a C&F scenario and the superiority of the proposed methods is discussed in terms of performance and complexity. Sec VI deals with convolutional lattices based on Construction D and their lattice decoding methods using the trellis structure of the underlying codes. Sec VII gives concluding remarks, including a discussion of further work required to apply the methods described to C&F, and to turbo lattices.

II. System Model

A point-to-point communication system exploiting nested lattice codes according to [13], and illustrated by Fig. 1, is investigated in this paper: the transmitter employs a lattice encoder which maps a message t to a Euclidean codeword c to be sent to the destination, i.e.,

$$c = [t \cdot G_\Lambda - u] \bmod -\Lambda \tag{1}$$

where u is a dithering signal, known to the transmitter and receiver, that is uniformly distributed in the Voronoi region of the coarse lattice. It should be noted that we assume hypercubic lattice in this paper (i.e., $\Lambda = q^N$ where q is the width of the hypercube per dimension and N is the dimension of the hypercube). G_Λ is the generator matrix of the lattice code that is obtained from a feed forward convolutional encoder according to Construction A and D in this paper. Note that the code rate is specified by the shaping lattice, i.e., the number of the lattice points inside the Voronoi region of the shaping lattice as well as the rate of the underlying convolutional code. Assuming M to be the number of lattice points inside the shaping region,

$$R = \frac{1}{N} \log_2 M \tag{2}$$

is defined as the code rate where N is the lattice dimension. The signal received at the destination is corrupted by Additive White Gaussian Noise (AWGN) as

$$v = c + n, \tag{3}$$

that is multiplied by the α coefficient to implement Minimum Mean Square Error (MMSE) estimation and the dither u is also added to the received signal[2],

$$y = \alpha v + u. \tag{4}$$

The signal y is then decoded by a lattice decoder and the transmitted message is recovered at the destination

$$\hat{c} = [\mathcal{Q}(y)] \bmod -\Lambda \tag{5}$$

$$\hat{t} = \mathcal{D}(\hat{c}) \tag{6}$$

where $\mathcal{Q}(\cdot)$ indicates a lattice quantizer/decoder and $\mathcal{D}$ maps a codeword to a message. Note that the $[\cdot] \bmod -\Lambda$ operation is a distributive operation and so one can rewrite (5) as

$$\hat{c} = \mathcal{Q}([y] \bmod -\Lambda) \tag{7}$$

which is equivalent to performing the modulo operation before lattice quantization. Indeed performing the modulo operation before or after lattice decoding/quantization does not affect the performance of the system and so a common trend in the literature is to apply existing lattice decoding algorithms, e.g., [20], [21], before the modulo operation since this leaves the structure of the lattice intact. However, we take a rather different approach and perform the modulo operation before lattice quantization as in (7).

[2]Please see [13] for detailed description about the role of MMSE estimator α and the dither u.

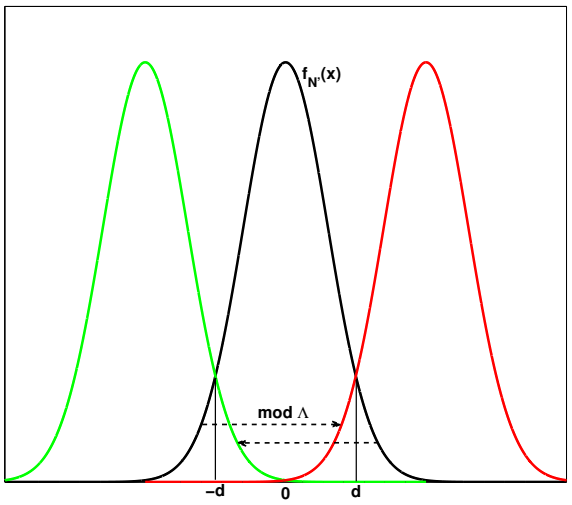

Fig. 2. A demonstration of $f_{N'}(x) = \frac{1}{4\eta_d}\text{erf}(\frac{x-\eta_d}{\sqrt{2}\alpha\sigma}, \frac{x+\eta_d}{\sqrt{2}\alpha\sigma})$ and modulo operation

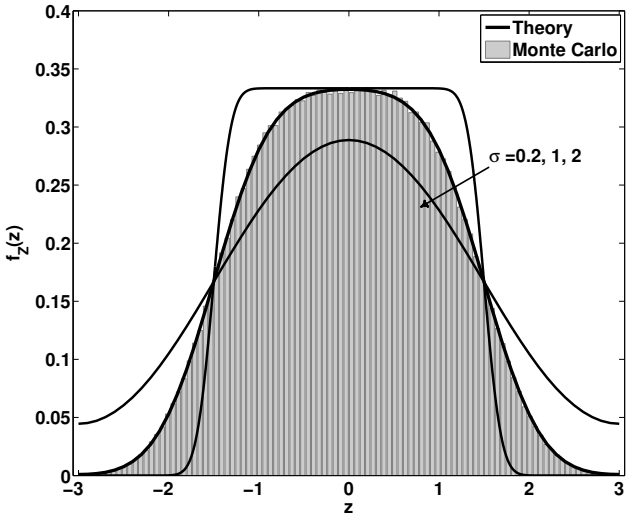

Fig. 3. pdf of modulo-lattice additive noise with $d = 3$, $\alpha = 0.5$ and various σ. For comparison, Monte Carlo simulations is provided for $\sigma = 1$.

III. Statistical Characteristics of Modulo Lattice Additive Noise

Statistical description of the overall receiver noise plays a key role in the design and theoretical analysis of communication systems; for instance, soft decoding algorithms, e.g., the BCJR algorithm, rely on the distribution of the additive noise in the receiver, which is usually modelled by the Gaussian distribution in conventional communication systems. However, in modulo-lattice channels wherein the receiver employs the modulo $-\Lambda$ operation before channel decoding, the additive noise is no longer Gaussian. Indeed, the additive noise lies inside the fundamental Voronoi region of the coarse lattice, and so, unlike the Gaussian noise, the modulo-Λ Gaussian noise domain doesn't range from $-\infty$ to $+\infty$ (i.e., $n_{\text{mod}} \notin (-\infty, +\infty)$). Following Erez *et al.* in [13], we will use the notation of "MLAN" (Modulo-Lattice Additive Noise) in this paper.

Considering that we assume a lattice with hypercubic shaping region in this paper based on Construction A/D, the $[\cdot]\text{mod} - \Lambda$ operation for an N dimensional lattice can be performed independently per dimension and so in the rest of this section we concentrate on deriving the statistical description of the noise in a single dimension. As n is normally distributed with zero mean and σ^2 variance, αn is also distributed normally. Moreover, the random variable $(1-\alpha)u$ is distributed uniformly in $[-d, +d)$ where $2d$ is the length of shaping hypercube, centred in origin. Consequently, the overall noise is the modulo-Λ of sum of two random variables of which one is distributed normally and the other is distributed uniformly:

$$Z = [\underbrace{(1-\alpha)u + \alpha n}_{N'}]\text{mod} - \Lambda \tag{8}$$

where the pdf of N' is derived as

$$f_{N'}(x) = \frac{1}{4\eta_d}\text{erf}(\frac{x-\eta_d}{\sqrt{2}\alpha\sigma}, \frac{x+\eta_d}{\sqrt{2}\alpha\sigma}) \tag{9}$$

with $\eta_d = d(1-\alpha)$; $\text{erf}(x, y) = \text{erf}(y) - \text{erf}(x)$ is the generalized error function. For a proof of (9), see Appendix A. Fig. 2 (black line) illustrates the pdf of N' that was derived in (9). A modulo-lattice operation is equivalent to mapping the area outside Voronoi region into inside the Voronoi region: for instance, in Fig. 2, the portion of the black curve in $(d, 3d)$ will be mapped inside the $(-d, d)$ region. It is clear from the figure that the portion of $f_{N'}(x)$ in $(d, 3d)$ is equal to the green curve at $(-d, d)$. Note that the green curve corresponds to the pdf of $(1-\alpha)u + \alpha n$ (as in (8)) where n follows the same distribution as in (8) and u is a random variable uniformly distributed in $(-3d, -d)$. Indeed, the the pdf of modulo-lattice noise in $(-d, d)$ is the sum of an infinite number of random variables with a pdf as in (37) with the centres located at $0, \pm 2d, \pm 4d, \cdots$. Consequently, the pdf of Z in (8) can be written as

$$f_Z(z) = \frac{1}{4\eta_d}\sum_{i=-\infty}^{\infty}\text{erf}(\frac{z-2di-\eta_d}{\sqrt{2}\alpha\sigma}, \frac{z-2di+\eta_d}{\sqrt{2}\alpha\sigma}). \tag{10}$$

Note that $f_Z(z)$ as derived in (10) will be used for lattice decoding of convolutional lattices using the BCJR algorithm in the following sections.

Truncation Error: Although the expression derived in (10) represents $f_Z(z)$ in closed-form, the infinite summation can be considered as a source of inconvenience in practice. Nevertheless, the infinite summation can be truncated with arbitrarily low truncation error. Note that $\lim_{i\to\infty}\text{erf}(\frac{z-2di+\kappa}{\sqrt{2}\alpha\sigma}, \frac{z-2di-\kappa}{\sqrt{2}\alpha\sigma}) = 0$, and so the significance of the expressions in (10) decreases as i increases. Fig.3 illustrates $f_Z(z)$ for various values of σ using the closed-form expression of (10) truncated at $i = \pm 2$. The result of Monte

Carlo simulations is also provided for comparison: it shows a perfect agreement between the theoretical plot and Monte Carlo simulations even for truncations as low as $i = \pm 2$.

IV. Constructing Lattices from Forward Error Correction Codes

Lattices with hypercubic Voronoi regions are particularly interesting because the mod-Λ operation in N dimensions can be performed independently in each dimension which results in considerable simplification of the problem. Since the complexity of specifying the Voronoi cell of a non-hypercubic lattice is unbounded in large dimensions [7], in a complexity-performance trade off, hypercubic lattices with lower complexity have been a potential candidate for practical purposes and so we focus on hypercubic lattices too. "Construction A" and "Construction D" are two well known and widely adopted lattice constructions that have hypercubic Voronoi regions. We adopt them from [1] for constructing convolutional lattices. Moreover, single and multi-layer Code Lattices analogous with Construction A and D, respectively. It will be observed that Code Lattices outperform their counterparts with considerable difference.

A. Preliminaries: Block Convolutional Codes

Assume a block convolutional code with rate k/N; the generator matrix of this block code is a $k \times N$ matrix that the rows of a generator matrix (i.e., basis vectors) are convolutional codewords generated by setting only one bit in the data vectors to one and the remaining bits to zero. For instance, let us assume that a codeword with length-N that is generated by a data-word as $[1, 0, \cdots, 0]$ is placed in the first row of the generator matrix as the first basis vector and similarly, the codeword that is generated by a data-word with a one in the i-th position is placed in the i-th row of the generator matrix. Consequently, <u>one</u> generator matrix of this $(7, 5)$ block convolutional code, based on the explanation above is

$$G_c = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & \\ & & & & \vdots & & & & & & \ddots & & \end{pmatrix}_{k \times N} \tag{11}$$

that will be used in next sections for constructing the lattices. It should be noted that the block convolutional codes are terminated by fixed zero bits in this paper.

B. Single Layer Convolutional Lattices: Construction A and Code Lattice

Construction A: Assume a (k, N, d) linear block code (block convolutional code in this paper) in $\mathbb{F}_q$ represented by $\mathcal{C} = \{c_0, c_1, \cdots, c_{M-1}\}$ with generator matrix G_c. Any vector $x = (x_1, \cdots, x_N)$ is a point of an N-dimensional lattice Λ_A, corresponding to codeword $c_i \in \mathcal{C}$ if and only if

$$[x] \bmod -\, q^N \triangleq c_i, \;\; c_i \in \mathcal{C}. \tag{12}$$

For instance, assuming $q = 2$, any vector x with even entries, is congruent to the codeword $c_0 = (0, \cdots, 0)$. In other words, any vector x with even entries is (i) a lattice point and (ii) represents the codeword c_0. In the following we discuss how the generator matrix of a convolutional lattice may be obtained from the generator matrix of the convolutional code.

Generator Matrix of Λ_A: For a given block convolutional code with rate k/N, the generator matrix of the "N-dimensional" convolutional lattice Λ_A constructed according to Construction A is

$$G_{\Lambda_A} = \begin{bmatrix} G_c \\ G \end{bmatrix}_{N \times N} \tag{13}$$

where G_c is the $k \times N$ generator matrix of the convolutional code (e.g., for a $(7, 5)$ convolutional code G_c is derived in (11)) and G is an $(N - k) \times N$ matrix with rows chosen from an $N \times N$ scaled identity matrix $qI_{N \times N}$ wherein the scaling parameter q is specified by the shaping lattice[3]. The matrix G must be chosen in a way that G_{Λ_A} is a full rank square matrix. Indeed, the role of G is to make G_{Λ_A} a rank N matrix. Minimum Euclidean distance of Λ_A ($d^{\Lambda_A}_{\text{min-u}}$) is

$$d^{\Lambda_A}_{\text{min-u}} = \min\{q, \sqrt{d^{\mathcal{C}}_{\text{min}}}\}. \tag{14}$$

where $d^{\mathcal{C}}_{\text{min}}$ is the minimum Hamming weight of the corresponding convolutional code. Note that for $q < \sqrt{d^{\mathcal{C}}_{\text{min}}}$, the error performance of the lattice is expected to be inferior to the error performance of the underlying block code because the performance of the lattice is bounded by the minimum Euclidean distance, which is smaller than the minimum Euclidean distance of the underlying convolutional code $d^{\Lambda_A}_{\text{min-u}} < \sqrt{d^{\mathcal{C}}_{\text{min}}}$.

Single Layer Code Lattice: The generator matrix of a block convolutional code G_c (size $k \times N$) can also be considered as the generator matrix of a "k-dimensional" lattice, which is called "single layer Code Lattice" ($\Lambda_{\mathcal{C}}$) in this paper. Note that although the dimension of the lattice is k, N coordinates are used to represent the lattice points in N dimensional space. The lattice points of $\Lambda_{\mathcal{C}}$ can be generated using $\mathbb{Z} \cdot G_c$ where the size of $\mathbb{Z}$ is $1 \times k$.

Minimum Euclidean distance of $d^{\Lambda_{\mathcal{C}}}_{\text{min-u}}$ is equal to the square of the Hamming distance of the underlying block convolutional code

$$d^{\Lambda_{\mathcal{C}}}_{\text{min-u}} = \sqrt{d^{\mathcal{C}}_{\text{min}}}. \tag{15}$$

<u>Remark 1</u>: It can easily be observed that $\Lambda_{\mathcal{C}} \subseteq \Lambda_A$ and so, clearly, $d^{\Lambda_A}_{\text{min-u}} \leq d^{\Lambda_{\mathcal{C}}}_{\text{min-u}}$. Indeed the extension of the $\Lambda_{\mathcal{C}}$ lattice to Λ_A is performed using sub-matrix G in (13).

For a better understanding the role of G in (13) (or equivalently, the extension of $\Lambda_{\mathcal{C}}$ to Λ_A), a simple two dimensional example is provided in the following: one generator matrix of a two dimensional hexagonal lattice on the $z = 0$ plane is $G_{\text{Hex}} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}$, nevertheless, the generator matrix of the lattice is not unique and, for instance, it can be represented

[3] In [1], Construction A is described only for $q = 2$, however, it does not necessarily require to be binary; in this paper we assume arbitrary q.

by another generator matrix as

$$G_{\text{Hex}} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \tag{16}$$

that uses three coordinates on the $x+y+z=0$ plane to represent a two dimensional lattice. The lattice generated by G_{Hex} in (16) is indeed the generator matrix of the two dimensional lattice $\Lambda_{\mathcal{C}}$ discussed above. In order to produce a lattice based on Construction A, one can concatenate, e.g., the row $[0\ 0\ 2]$ with (16) and obtain

$$G_{\Lambda_A} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 2 \end{bmatrix}. \tag{17}$$

Note that the third row in (17) copies the two dimensional hexagonal lattice to the third dimension parallel to $x+y+z=0$ plane and generates the three dimensional Λ_A lattice. It's worth to mention that, in this example, the $d^{\mathcal{C}}_{\text{min}}$ is the MED of the hexagonal lattice and so $d^{\mathcal{C}}_{\text{min}} = 2$. Consequently $d^{\Lambda_A}_{\text{min-u}} = \min(q, \sqrt{d^{\mathcal{C}}_{\text{min}}}) = \min(2, \sqrt{2}) = \sqrt{2}$ which is bounded by the MED of the code, hence, the error performance of G_{Hex} in (16) and G_{Λ_A} in (17) are expected to be rather similar; however, for many advanced channel codes, the MED of the code is larger than q and so the error performance of the $\Lambda_{\mathcal{C}}$ lattice is better than the error performance of Λ_A lattice. This will be confirmed by simulation later in the paper.

Remark 2: A *transmitter* exploiting lattices (based on Construction A) as the channel code, generates only the lattice points inside the Voronoi region of the shaping lattice q^N as follows: assuming that $\boldsymbol{u}_{1\times N}$ is the data vector that represents M messages, one can write

$$u_i \in \begin{cases} \{0, 1, \cdots, q-1\}, & \text{for } i \le k \\ \{0\}, & \text{for } i > k, \end{cases} \tag{18}$$

consequently, $\boldsymbol{u} = [\boldsymbol{u}_{\text{data}}\ \boldsymbol{u}_{\text{duplicate}}]$ where $\boldsymbol{u}_{\text{data}}$ is a $1\times k$ vector that is specified by the first row of (18); $\boldsymbol{u}_{\text{duplicate}}$ is a $1\times(N-k)$ vector that does not carry any information. Note that the transmitter exploits $[\boldsymbol{u}\cdot G_{\Lambda_A}] \text{mod} - q^N$ for assigning a lattice point to a message and so any value assigned to $\boldsymbol{u}_{\text{duplicate}}$ will be discarded by $[\cdot] \text{mod} - q^N$ operation. Note that one can also generate lattice points generated by $[\boldsymbol{u}\cdot G_{\Lambda_A}] \text{mod} - q^N$ using $[\boldsymbol{u}_{\text{data}}\cdot G_{\Lambda_{\mathcal{C}}}] \text{mod} - q^N$ and so the lattice points inside q^N hypercube are common lattice points between Λ_A and $\Lambda_{\mathcal{C}}$. Note that it is important to distinguish between the two lattices because "*lattice decoding*" using Λ_A or $\Lambda_{\mathcal{C}}$ can lead to different error performance and so one should use appropriate lattice (the lattice with the larger minimum distance) for the purpose of lattice decoding.

C. Decoding Single Layer Convolutional Lattice Codes using Trellis Structure of the Code

Although lattice decoding (indeed lattice quantization) is usually considered to be less complex than ML decoding because of the structure of the lattice, a practical "universal lattice decoding" algorithm with reasonable complexity is still a hot research topic in the field. There are several lattice decoding algorithms proposed in the literature [1], [6], [7], [22] , however, the algorithms are only applicable in very low dimensions; for example, in [7] it is clarified that the proposed algorithm has been examined for decoding lattices up to 32 dimensions. Considering that transmission rates close to capacity can be approached only by lattices with high dimension, the existing universal lattice decoding algorithms do not seem to be very appealing in practice.

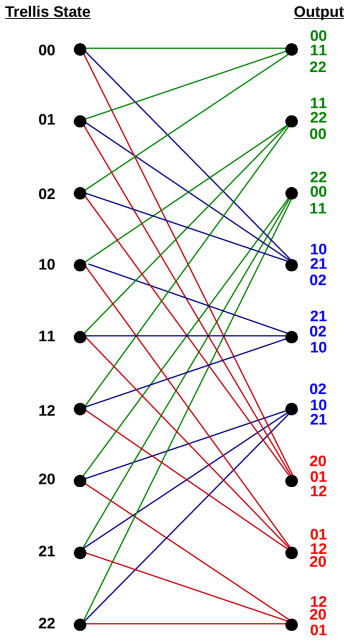

Fig. 4. Trellis representation of $(7,5)$ convolutional lattice with $q=3$. Input is in $\{0,1,2\}$ and output is in $\{0,1,2\ (\text{or } -1)\}$. Green lines represents the transition corresponding to input 0, blue corresponds to input 1 and red for input equal to 2.

Apart from the universal lattice decoders, several lattice decoding algorithms have been proposed for certain lattices obtained using particular FEC codes; for instance, lattice decoders based on the sum-product algorithm have been proposed in [23], [24] for Low Density Lattice Codes (LDLC). Likewise, in this paper, we are *not* interested in a universal lattice decoding algorithm for an arbitrary lattice but in lattice decoding of convolutional lattices that are obtained using convolutional codes.

Before we continue with lattice decoding of convolutional lattices, let us concentrate on ML/MAP decoding of convolutional lattices and notice that ML and MAP decoding algorithms with reasonable complexity exist (i.e., Viterbi and BCJR, respectively).

ML and MAP Decoding of Convolutional Lattices: For ML/MAP decoding of convolutional lattices, one can simply resort to the trellis structure of the corresponding convolutional code: for instance, assuming $q=2$ and preserving the order of the basis vectors in the generator matrix of the lattice according to Section IV-A, the transmitted lattice points are exactly equal to the corresponding binary convolutional code and so one can easily employ the trellis structure of the convolutional code for Viterbi/BCJR decoding of the convolutional lattice. For shaping hypercubes $q>2$, the trellis structure is not hard to derive. As an example, assume $q=3$,

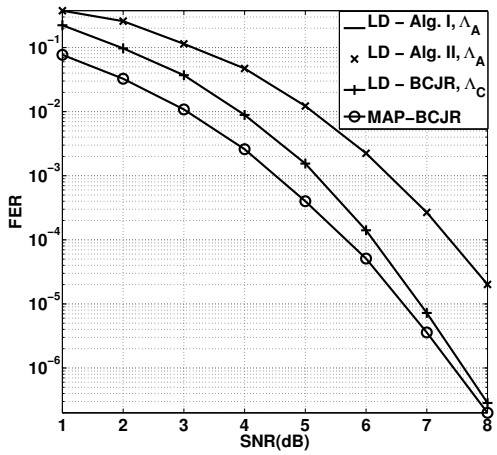

Fig. 5. Frame Error Rate for $(7, 5)$ convolutional lattice with $q = 2$ and dimension $N = 20$.

and so, the lattice encoding is performed in $\mathbb{F}_3$. Assuming a shift register based convolutional encoder which performs operations in $\mathbb{F}_3$, the trellis structure is illustrated by Fig. 4 for a $(7, 5)$ convolutional code where the green arrows represent transitions corresponding to input $u_i = 0$, blue and red arrows show transitions corresponding to $u_i = 1$ and $u_i = 2$, respectively. Note that the overall number of the codewords inside the shaping hypercube is 3^k and so the code rate is

$$R = \frac{1}{n}\log_2 3^k = \frac{1}{2}\log_2 3. \tag{19}$$

It is important to clarify that using the trellis structure of the underlying convolutional code for ML/MAP decoding of the lattice, we use the Code Lattice ($\Lambda_{\mathcal{C}}$) for decoding and not the Construction A lattice (Λ_A). Note that $d_{\min}^{\Lambda_{\mathcal{C}}} \geq d_{\min}^{\Lambda_A}$ and consequently decoding on $\Lambda_{\mathcal{C}}$ outperforms decoding on Λ_A.

Lattice Decoding of Convolutional Lattices: As discussed earlier in (5) and (7), performing the modulo operation before or after lattice quantization will not affect the performance of the system; therefore, for the purpose of lattice decoding, we perform the modulo operation before lattice quantisation which consequently maps the entire space to the inside of the Voronoi region of the shaping lattice which includes only the lattice points that can actually be transmitted from the source. Therefore, one can exploit the trellis structure of the underlying convolutional lattice for lattice decoding (e.g., Viterbi or BCJR). Note that the only difference with the ML/MAP decoding discussed earlier is the $[\cdot] \bmod -\, q^N$ operation and so we refer to this as "Lattice decoding using Viterbi/BCJR" algorithm (where the noise is MLAN) whereas ML/MAP decoding refers to conventional Viterbi/BCJR decoding algorithms without the $[\cdot] \bmod -\, q^N$ operation (i.e., with Gaussian noise). Considering that the BCJR algorithm requires the statistical description of the overall noise, we exploit the pdf of MLAN derived in (10) for calculating the state transition probabilities of the BCJR algorithm.

Fig. 5 illustrates the Frame Error rate (FER) obtained using computer simulations for a $(7, 5)$ convolutional lattices

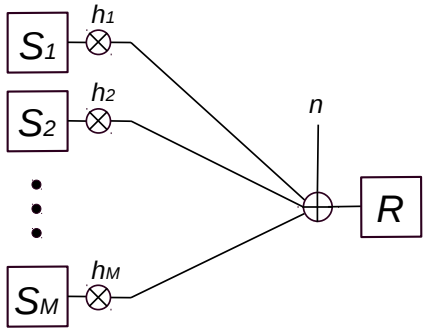

Fig. 6. System Model: Compute and Forward.

with $q = 2$ (i.e., $\mathbb{F}_2$) on a lattice with dimension equal to 20. Assuming Gaussian noise and modulo operation before BCJR decoding (i.e., Lattice Decoding (LD) using BCJR algorithm), FER is shown by the bold line marked with (+); for comparison two universal lattice decoding algorithms from [5] and [6] are indicated with LD-Alg. I and LD-Alg. II, respectively. It is clear that the proposed BCJR based lattice decoding of convolutional lattices outperforms conventional lattice decoding methods with more than 1.5 dB difference. Note that in Fig. 5 we are forced to perform computer simulations in low dimension (20 dimension) because existing universal lattice decoders, i.e., LD-Alg. I and LD-Alg. II from [5] and [6], that are used as a benchmark for comparison, are practically feasible only in low dimensions. Fig. 5 also shows the BCJR decoding of the convolutional lattice (*without* performing modulo operation, say MAP decoding). Note that MAP decoding outperforms lattice decoding because of the poor error performance of lattice decoding[4], however, clearly both the curves converge at high SNR, as expected.

In the following theorem, the advantage of $\Lambda_{\mathcal{C}}$ over Λ_A that leads to a superior error performance as illustrated in Fig. 5 is discussed.

Theorem 1. *Considering error rate as a performance benchmark, lattice decoding of single layer Code Lattice ($\Lambda_{\mathcal{C}}$) outperforms lattice decoding of the corresponds Construction A lattice (Λ_A).*

Proof: Comparing the lattice decoders in Fig. 5 puts forward the question of why the proposed lattice decoder outperforms existing universal lattice decoders? There are indeed two main reasons for this that are explained in the following:

- The proposed lattice decoding using the BCJR algorithm is performed over the Code Lattice ($\Lambda_{\mathcal{C}}$) whereas lattice decoding using universal lattice decoders proposed by [5], [6] is performed over Λ_A. The minimum Euclidean distance of $\Lambda_{\mathcal{C}}$ is equal to the minimum Euclidean distance of the underlying convolutional code, i.e., $d_{\min}^{\Lambda_{\mathcal{C}}} = \sqrt{d_{\min}^{\mathcal{C}}}$, while $d_{\min}^{\Lambda_A} \leq \sqrt{d_{\min}^{\mathcal{C}}}$. Hence, lattice decoding on $\Lambda_{\mathcal{C}}$ using the BCJR algorithm outperforms existing universal lattice

[4]Please see [13]–[16] for a thorough discussion about the error exponent of MLAN channels.

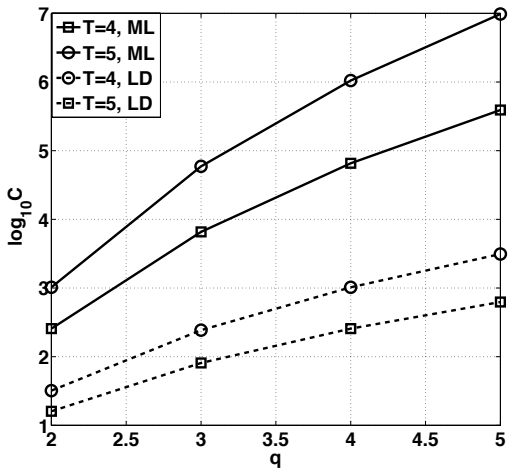

Fig. 7. Complexity of ML and Lattice decoding for various values of constraint length T and assuming five users ($M = 5$).

decoders that perform decoding on Λ_A lattice. Note that universal lattice decoding algorithms proposed, e.g., in [5]–[7] require generator matrix in square form and so we are forced to use the Λ_A lattice with square generator matrix for decoding.

- In a Λ_A lattice which is generated from a $k \times N$ convolutional code, the dimension of the lattice increases from k to N (using G sub-matrix in (13)) and consequently the number of adjacent lattice points that can erroneously be decoded increases.

■

As discussed in earlier sections, considering that a lattice decoder is, in general, outperformed by an ML decoder, application of lattice decoders in practice does not seem to be a justifiable choice in point-to-point communication systems, however, the idea of obtaining convolutional lattices, in this paper, was initially motivated by C&F relaying [19]. In order to validate the usefulness of lattice decoding, the next section, we consider a C&F relaying where lattice decoding is the method of choice due to its manageable complexity in practical systems.

V. Application of Convolutional Lattices in Compute and Forward

In this section, we use C&F relaying to validate the usefulness of the proposed lattice decoder by comparing the complexity of the proposed lattice decoder with an ML/MAP decoder. Assume a C&F relaying systems wherein multiple source nodes transmit their data simultaneously towards a relay node. For instance, Fig. 6 illustrates the Multiple Access Channel (MAC) phase of a relaying system wherein the source nodes employ convolutional lattice codes as the FEC code. The relay node performs channel decoding (whether ML or lattice decoding); upon decoding the resultant lattice point in the relay, $[\cdot] \bmod -\, q^N$ operation is performed as network coding and a new lattice point is then sent to the intended destination nodes (see e.g., [19] for detailed description about C&F). Assuming the source nodes to apply the same convolutional codes, the complexity of the relay node employing the proposed lattice decoder is considerably lower than the complexity of the equivalent ML/MAP decoder. Let us define the complexity of the proposed lattice decoder for convolutional lattices as the number of trellis states

$$C = q^{T-1} \quad \text{trellis states} \tag{20}$$

where C is the measure of complexity and T is the constraint length of corresponding convolutional code; note that the complexity of the lattice decoder is independent of the number of the source nodes whereas assuming M source nodes, the complexity of ML/MAP decoder is $C = q^{M(T-1)}$ trellis states. Consequently, as illustrated by Fig. 7, the complexity of the ML decoder is indeed much more than the lattice decoder.

As an example to evaluate the error performance of a C&F system, in Fig. 6, assume a relaying system with two transmit nodes that both exploit convolutional lattice of dimension 1000 based on $(7, 5)$ code and shaping lattice of $q = 3$ (i.e., hypercube of 3^{1000} that results in rate $\frac{1}{2}\log_2 3$). The channel corresponding to first user is set to one ($h_1 = 1$) and various values are assumed for the second user's channel as $h_2 = 1, 1.2$ and 1.4. Fig. 8 shows the FER of this C&F system with proposed lattice decoding used in the relay node: as expected, the self-noise (see, e.g., [18] or [19] for more discussion about the self-noise) degrades the performance of the system due to non-integer values of h_2, however, from a complexity point of view, the proposed lattice decoder using BCJR algorithm performs decoding with only $3^2 = 9$ trellis states that can be implemented on a real hardware, whereas an ML/MAP decoder will have $3^{2\times 2} = 81$ trellis states. This demonstrates the advantage of the proposed lattice decoder in certain communication systems like C&F. Note that since the complexity of an ML/MAP decoder grows exponentially with the number of the users, practical implementation of it is indeed impossible with moderate and large number of users.

VI. Encoding and Decoding Multilayer Convolutional Lattices

Lattices based on Construction D (Λ_D) and what we refer to as "multilayer Code Lattices ($\Lambda_{\mathcal{C}}$)[5]" are the two lattice constructions discussed in this section. However, before we continue with the definition of Construction D and the construction of lattices from convolutional codes exploiting the Construction D template, we would like to focus also on conventional multilevel coding [25] techniques which are referred to as Construction by Code Formula (CCF). This is discussed in the next subsection. Note that we are interested in CCF because both CCF and Construction D are usually regarded equivalent in the literature (e.g., see [26]–[28]).

A. Multilevel Codes or Construction by Code Formula

Assume a family of a binary linear codes in which

$$\mathbb{F}_2^N \supseteq \mathcal{C}_1 \supseteq \mathcal{C}_2 \cdots \supseteq \mathcal{C}_a \tag{21}$$

[5]That is analogous to single layer Code Lattice discussed in Section IV-B.

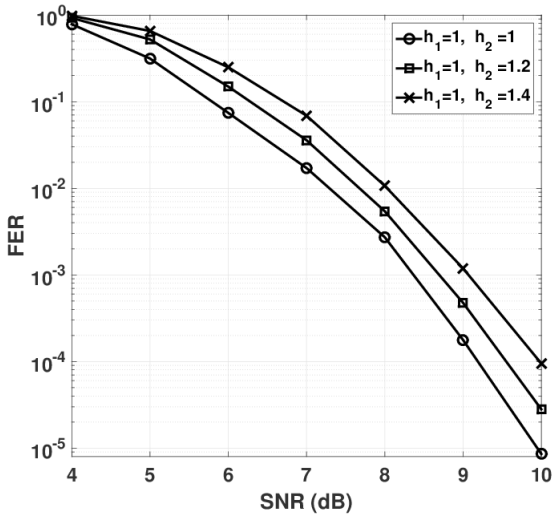

Fig. 8. Frame Error Rate for a C&F system with two transmit nodes exploiting $(7,5)$ convolutional lattice with $q=3$, $h_1=1$ and various values for h_2.

with $\mathcal{C}_i$ as a $[k_i, N, d_i]$ linear block code. A code based on CCF will be defined as

$$\mathcal{C}_{\text{CCF}} = \psi_1(\mathcal{C}_1) + \psi_2(\mathcal{C}_2) + \cdots + \psi_a(\mathcal{C}_a) \tag{22}$$

where $\psi(\cdot)$ is a map from $\mathbb{F}_2^N$ to $\mathbb{R}^N$ in which $\psi_i(\boldsymbol{x}) = \frac{\boldsymbol{x}}{2^{i-1}}$ where $\boldsymbol{x} \in \mathcal{C}_i$. For instance, assuming $a=3$, one can write $\mathcal{C}_{\text{CCF}} = \frac{1}{4}\mathcal{C}_3 + \frac{1}{2}\mathcal{C}_2 + \mathcal{C}_1$ for the resultant code[6]. The code rate R_{CCF} of CCF is $R_{\text{CCF}} = \sum_{i=1}^{a} R_{\mathcal{C}_i}$. The desired aspects of CCF, among the others, is that encoding (and decoding) $\mathcal{C}_{\text{CCF}}$ can be performed using the conventional encoding (and decoding) methods used for the underlying code $\mathcal{C}_i$. For instance, one can use Viterbi or BCJR algorithm for decoding a convolutional $\mathcal{C}_{\text{CCF}}$, wherein the receiver decodes the inner layer $\mathcal{C}_a$ first and exploits it as *a priori* knowledge passed to the decoder which decodes the layer corresponding to $\mathcal{C}_{a-1}$. This multi-stage decoding algorithm continues until all the layers are decoded. Note that multilevel codes and multi-stage decoding algorithms are extensively studied in the literature (see, e.g., [25]) and so we will adopt them in the following for encoding and decoding lattices based on Construction D.

In the following, taken from [1], we will define Construction D and will explain, using a counterexample, that Construction D and CCF can be different.

B. Construction D

Assume a family of a binary linear codes in which

$$\mathbb{F}_2^N \supseteq \mathcal{C}_1 \supseteq \mathcal{C}_2 \cdots \supseteq \mathcal{C}_a \tag{23}$$

with $\mathcal{C}_i$ as a $[k_i, N, d_i]$ linear block code where $d_i \geq 4^i/\gamma$ in which $\gamma = 2$ or 4. Choose N basis vectors of $\mathbb{F}_2^N$ (i.e., $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_N\}$) wherein the set of $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_{k_i}\}$ spans $\mathcal{C}_i$. Also assume that $\psi(\cdot)$ is a map from $\mathbb{F}_2^N$ to $\mathbb{R}^N$ in which $\psi_i(\boldsymbol{x}) = \frac{\boldsymbol{x}}{2^{i-1}}$ where $\boldsymbol{x} \in \mathcal{C}_i$. A lattice based on Construction D contains all vectors of the form

$$\Lambda_D = \sum_{i=1}^{a} \sum_{j=1}^{k_i} \alpha_j^{(i)} \psi_i(\boldsymbol{b}_j) \; + \; (2\mathbb{Z})^N \tag{24}$$

where $\alpha_k^{(j)} \in \{0,1\}$. Consequently, Λ_D is an N dimensional lattice with hypercube fundamental Voronoi region. Assuming $\mathcal{C}_D$ to be a code consisting of all lattice points inside the $(-1,1]^N$ hypercube of Λ_D lattice, any point/vector $\boldsymbol{x} = (x_1, \cdots, x_N) \in \mathbb{R}^N$ is a lattice point congruent to codeword c_i if and only if

$$[x] \text{mod} - 2^N \triangleq c_i, \;\; c_i \in \mathcal{C}_D. \tag{25}$$

Note that as both the CCF and Construction D are defined based on the $\psi(\cdot)$ function, they are usually considered to be equivalent in the literature (e.g., see [26]–[28]), however, in order to disprove this conjecture, a counterexample is provided in the following which shows that CCF does not necessarily result in a lattice construction.

Counterexample: Assume two nested binary codes $\mathcal{C}_1 \supseteq \mathcal{C}_2$ with $G_1 = \left[\begin{smallmatrix}1100\\1010\\1001\end{smallmatrix}\right]$ and $G_2 = [\begin{smallmatrix}1100\\1010\end{smallmatrix}]$. The codebook of CCF inside the 2^4 hypercube obtained from $\mathcal{C}_{\text{CCF}} = \frac{1}{2}\mathcal{C}_2 + \mathcal{C}_1$ is

$$\begin{aligned}\mathcal{C}_{\text{CCF}} = \Big\{ &[0\;0\;0\;0]\,[0\;0\;1\;1]\,[0\;1\;0\;1]\,[0\;1\;1\;0]\\ &[1\;0\;0\;1]\,[1\;0\;1\;0]\,[1\;1\;0\;0]\,[1\;1\;1\;1]\\ &[0.5\;0\;0.5\;0]\,,[0.5\;0\;1.5\;1]\,,[0.5\;0.5\;0\;0]\,,[0.5\;0.5\;1\;1]\\ &[0.5\;1\;0.5\;1]\,,[0.5\;1\;1.5\;0]\,,[0.5\;1.5\;0\;1]\,,[0.5\;1.5\;1\;0]\\ &[1.5\;0\;0.5\;1]\,,[1.5\;0\;1.5\;0]\,,[1.5\;0.5\;0\;1]\,,[1.5\;0.5\;1\;0]\\ &[1.5\;1\;0.5\;0]\,,[1.5\;1\;1.5\;1]\,,[1.5\;1.5\;0\;0]\,,[1.5\;1.5\;1\;1]\\ &[0\;0.5\;0.5\;0]\,,[0\;0.5\;1.5\;1]\,,[0\;1.5\;0.5\;1]\,,[0\;1.5\;1.5\;0]\\ &[1\;0.5\;0.5\;1]\,,[1\;0.5\;1.5\;0]\,,[1\;1.5\;0.5\;0]\,,[1\;1.5\;1.5\;1]\Big\}. \end{aligned} \tag{26}$$

The generator matrix of Construction D is[7]

$$G_{\Lambda_D} = \left[\begin{smallmatrix}\frac{1}{2}&\frac{1}{2}&0&0\\ \frac{1}{2}&0&\frac{1}{2}&0\\ 1&0&0&1\\ 2&0&0&0\end{smallmatrix}\right], \tag{27}$$

with which, the lattice points inside the 2^4 hypercube are

$$\begin{aligned}\mathcal{C}_D = \Big\{ &[0\;0\;0\;0]\,[0\;0\;1\;1]\,[0\;1\;0\;1]\,[0\;1\;1\;0]\\ &[1\;0\;0\;1]\,[1\;0\;1\;0]\,[1\;1\;0\;0]\,[1\;1\;1\;1]\\ &[0.5\;0\;0.5\;0]\,[0.5\;0\;1.5\;1]\,[0.5\;0.5\;0\;0]\,[0.5\;0.5\;1\;1]\\ &[0.5\;1\;0.5\;1]\,[0.5\;1\;1.5\;0]\,[0.5\;1.5\;0\;1]\,[0.5\;1.5\;1\;0]\\ &[1.5\;0\;0.5\;1]\,[1.5\;0\;1.5\;0]\,[1.5\;0.5\;0\;1]\,[1.5\;0.5\;1\;0]\\ &[1.5\;1\;0.5\;0]\,[1.5\;1\;1.5\;1]\,[1.5\;1.5\;0\;0]\,[1.5\;1.5\;1\;1]\\ &[0\;0.5\;0.5\;1]\,[0\;0.5\;1.5\;0]\,[0\;1.5\;0.5\;0]\,[0\;1.5\;1.5\;1]\\ &[1\;0.5\;0.5\;0]\,[1\;0.5\;1.5\;1]\,[1\;1.5\;0.5\;1]\,[1\;1.5\;1.5\;0]\Big\}. \end{aligned} \tag{28}$$

Careful comparison of (26) and (28) reveals that the two last rows in (26) and (28) are different and so, one can

[6]Alternatively, one can write $\mathcal{C}_{\text{CCF}} = \mathcal{C}_3 + 2\mathcal{C}_2 + 4\mathcal{C}_1$ too.

[7]Note that obtaining generator matrix of a lattice based on Construction D will be explained in further detail in the following, however, in order to validate the difference between Construction D and CCF, we use it in this example without a proof.

easily conclude that CCF and Construction D are not *necessarily* equivalent. Furthermore, although $[1.5\ 1.5\ 1\ 1]$ and $[0\ 1.5\ 1.5\ 0]$ are points of $\mathcal{C}_{\text{CCF}}$ in (26) (the last vectors in row six and seven), their mod-2 sum

$$\big[[1.5\ 1.5\ 1\ 1]+[0\ 1.5\ 1.5\ 0]\big] \bmod -2^4 = [1.5\ 1\ 0.5\ 1] \quad (29)$$

does not belong to $\mathcal{C}_{\text{CCF}}$ in (26) and so, clearly, in this example, the CCF does not generate lattice points. In general, the CCF does not necessarily generate a lattice, however, for the particular case of convolutional codes, we introduce an approach with which lattice points of Construction D are generated using CCF. ■

Note that on the one hand we are interested in constructing a lattice from (convolutional) codes based on Construction D and, on the other hand, we would like to make Construction D and CCF equivalent because then we can exploit existing and practically feasible decoding algorithms of CCF for decoding convolutional lattices that are constructed based on Construction D, else, as discussed in the context of construction A, *universal lattice decoders* are not interesting from a practical point of view at high dimensions.

In the following, we focus on deriving the generator matrix of a lattice constructed from convolutional codes according to Construction D and its equivalent CCF. For this, we first neglect the minimum distance criterion of Construction D definition in the following (i.e., the $d_i \geq 4^i/\gamma$ criterion); later on, we will discuss methods for ensuring the minimum distance criterion is fulfilled which indeed can improve the performance of the code[8].

a) No Minimum Distance Criterion: The generator matrix of the convolutional lattice will be obtained as follows: the first k_a basis vectors[9] multiplied by $1/2^{a-1}$ form the first k_a rows of the generator matrix; the rows from k_a+1 to k_{a-1} are obtained by multiplying the k_a+1 to k_{a-1} rows of the convolutional code generator matrix G_c to $1/2^{a-2}$. Similarly one can obtain all the rows of the generator matrix of the lattice using the $\psi(\cdot)$ mapping that corresponds to the associated convolutional code. The remaining $(N-k)$ rows are chosen from the rows of a $2I_{N\times N}$ matrix in such a way that the generator matrix of the lattice has rank N.

In the following, the generator matrix of a convolutional lattice based on Construction D is further discussed using an example.

Example 1: Assume three $(7,5)$ convolutional codes as $\mathbb{F}_2^N \supseteq \mathcal{C}_1 \supseteq \mathcal{C}_2 \supseteq \mathcal{C}_3$ and let $G_{\mathcal{C}_i}$ be the generator matrix of a $(7,5)$ convolutional code as derived in (11). Note that since we assume nested codes, we mean that all $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ contain equal length codewords (and data words) and so it implies that the data words of the sub-codes are zero padded (ZP) to make the length of the data vectors of the sub-codes equal to the length of the data vector of the parent code $\mathcal{C}_1$. For instance assume $k_3=1$, $k_2=2$, $k_1=4$; therefore, the generator matrix of the the three nested codes is as follows: $G_{\mathcal{C}_3}=[\boldsymbol{b}_1]_{1\times 8}$, $G_{\mathcal{C}_2}=[\boldsymbol{b}_1;\boldsymbol{b}_2]_{2\times 8}$, $G_{\mathcal{C}_1}=[\boldsymbol{b}_1;\boldsymbol{b}_2;\boldsymbol{b}_3;\boldsymbol{b}_4]_{4\times 8}$ where $\boldsymbol{b}_i$ is the i-th row of (11). The generator matrix of the lattice constructed according to Construction D is

$$G_{\Lambda_D} = \begin{bmatrix} \frac{1}{4}\boldsymbol{b}_1 \\ \frac{1}{2}\boldsymbol{b}_2 \\ 1\boldsymbol{b}_3 \\ 1\boldsymbol{b}_4 \\ G \end{bmatrix}_{8\times 8}. \quad (30)$$

Note that the coefficient $\frac{1}{4}$ for $\boldsymbol{b}_1$, the coefficient $\frac{1}{2}$ for $\boldsymbol{b}_2$ and 1 for $\boldsymbol{b}_3$ and $\boldsymbol{b}_4$ represent the $\psi_i(\cdot)$ function in (24) and G contributes the $(2\mathbb{Z})^N$ part of (24). All the lattice points inside the $(-1,1]^N$ hypercube are the codewords of the Construction D convolutional lattice, and are obtained using the $[\boldsymbol{u}\cdot G_{\Lambda_D}] \bmod -2^N$ operation where $\boldsymbol{u}\in\mathbb{Z}$.

In order to take advantage of decoding algorithms of CCF, in the following, a method is introduced by which Construction D and CCF are equivalent.

Construction D using CCF: In the above example, there are three nested codes, contributing in encoding four bits (say $\{d_1,d_2,d_3,d_4\}$ corresponding to basis vectors $\{\boldsymbol{b}_1,\boldsymbol{b}_2,\boldsymbol{b}_3,\boldsymbol{b}_4\}$, respectively). Consequently, one can say that d_1 is encoded by $\mathcal{C}_3$, d_2 by $\mathcal{C}_2$ and d_3,d_4 by $\mathcal{C}_1$. In order to generate Construction D lattice points inside the 2^4 hypercube, similar to multilevel codes (or CCF), one can write the equivalent transmitter side generator matrix as

$$G_{\Lambda_D}^{\text{TX-eqv}} = \begin{bmatrix} \begin{bmatrix} \frac{1}{4}\boldsymbol{b}_1 \end{bmatrix} \\ \begin{bmatrix} \frac{1}{2}\boldsymbol{b}_1 \\ \frac{1}{2}\boldsymbol{b}_2 \end{bmatrix} \\ \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_4 \end{bmatrix} \end{bmatrix}, \quad (31)$$

however, set the data bits corresponding to the basis vectors indicated by the red color to zero, i.e.,

$$\boldsymbol{d}_{\text{eq}} = [\underbrace{d_1}_{\mathcal{C}_3}, \underbrace{0,d_2}_{\mathcal{C}_2}, \underbrace{0,0,d_3,d_4}_{\mathcal{C}_1}]. \quad (32)$$

Consequently, $\left[\boldsymbol{d}_{\text{eq}}\cdot G_{\Lambda_D}^{\text{TX-eq}}\right] \bmod -2$ will generate the same lattice points inside the 2^4 hypercube that will be generated by $[\boldsymbol{d}\cdot G_{\Lambda_D}] \bmod -2^4$ where $\boldsymbol{d}=[d_1,d_2,d_3,d_4]$.

Note that the upper sub-matrix in (31) corresponds to $\mathcal{C}_3$, the sub-matrix in the middle corresponds to $\mathcal{C}_2$ and the bottom one corresponds to $\mathcal{C}_1$; hence, instead of using matrix multiplication for generating lattice points (codewords), one can use conventional shift register based encoders according to Fig. 9 in the transmitter where

$$\boldsymbol{d}_3 = [d_1, \underbrace{0,0,0}_{\text{ZP}}] \quad (33a)$$

$$\boldsymbol{d}_2 = [0, d_2, \underbrace{0,0}_{\text{ZP}}] \quad (33b)$$

$$\boldsymbol{d}_1 = [0,0,d_3,d_4]. \quad (33c)$$

[8]Note that the main reason for neglecting the minimum distance criterion is due to a lack of *nested* convolutional codes that fulfil the minimum distance criterion; therefore, many papers (e.g., [3], [4], [29]) relax this criterion.

[9]The basis vectors are obtained according to the description in Section IV-A.

Consequently, $[\boldsymbol{d} \cdot G_{\Lambda_D}] \bmod -2$, $\left[\boldsymbol{d}_{\text{eq}} \cdot G_{\Lambda_D}^{\text{TX-eq}}\right] \bmod -2$ and Fig. 9 will generate the same lattice points inside the 2^4 hypercube.

b) Ensuring Minimum Distance Criterion: So far, we have relaxed the minimum distance criterion of the original definition of Construction D. The relaxation of the minimum distance criterion was, in part, to focus on constructing lattices using conventional convolutional codes; however, the literature usually ignores the minimum distance criterion(e.g., see [3], [23], [29]) because it is hard to find nested codes that fulfil this criterion. In particular, in the case of convolutional codes, the minimum distance of a code is fixed; for instance, it is well known that the minimum distance of the $(7, 5)$ convolutional code that was used in the above example is equal to five.

In the previous part, we applied a naive method of zero padding to have data words of equal length in all convolutional codes (e.g., see (33)), however, one can perform repetition of the uncoded data bits instead of simply zero padding which indeed can result in increasing the minimum distance of the code.

For clarity of explanation, let us begin with the same $(7, 5)$ convolutional code with the generator matrix derived in (11). Note that (11) is only one of the generator matrices of the $(7, 5)$ code; each basis vector in (11) can be replaced by another basis vector. For instance, the basis vector in the first row of (11) (i.e., $\boldsymbol{b}_1 = [11101100\cdots]$) can be replaced by a new $\boldsymbol{b}_1$ where $\boldsymbol{b}_1 = [110101110\cdots]$ that is a codeword generated by a data vector with the first two bits set to 1 and the rest of the bits set to zero. Note that in *Example 1* where only one bit d_1 is transmitted by the inner code $\mathcal{C}_3$, one can repeat d_1 instead of zero padding: indeed, substituting $\boldsymbol{b}_1 = [11101100\cdots]$ with $\boldsymbol{b}_1 = [110101110\cdots]$ in (30) is equivalent to transmitting $[d_1, d_1, 0, 0]$ from $\mathcal{C}_3$. Moreover, considering that $\mathcal{C}_3$ produces only two codewords (all zero and $[110101110000]$), clearly, the minimum distance of the code has increased to 6. Note that codes with larger minimum distance can be produced by different repeating patterns for different convolutional codes: for instance, repeating d_1 according to $[d_1, 0, 0, d_1]$ is equivalent to replacing $\boldsymbol{b}_1 = [11101100\cdots]$ with $\boldsymbol{b}_1 = [11101111101100]$ and so $\mathcal{C}_3$ will produce two codewords (all zero and $[11101111101100]$) that have minimum distance equal to 10.

Note that the repeating pattern depends on the convolutional code, however, for the particular example of the $(7, 5)$ convolutional code, repeating $[d_i 00]$ along the data word, *instead of zero padding*, will generate codewords with maximum MED.

C. Decoding Multilayer Convolutional Lattice Codes

One obvious way of lattice decoding convolutional lattices obtained using Construction D is to employ the well known universal lattice decoders in combination with the original generator matrix of Λ_D as obtained, e.g., in (30). However, as observed in Section IV-C, the performance will be poor due to the edge lattice points and the MED of the lattice which is upper bounded by 2. Another solution is to decode the lattice in k dimensions instead of N and obtain better performance; this is discussed in the following. The following corollary is provided as a result of theorem 1 for multilayer convolutional lattices:

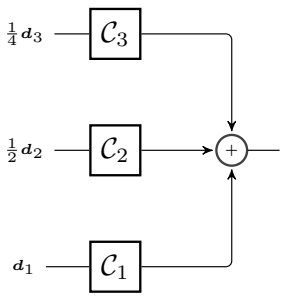

Fig. 9. Equivalent Construction D encoder using Conventional FEC encoders

Corollary 1. *Considering error rate as a performance benchmark, lattice decoding of multilayer Code Lattice ($\Lambda_{\mathcal{C}}$) outperforms lattice decoding of the corresponds Construction D lattice (Λ_D).*

Proof: The proof of single layer convolutional lattices in theorem 1 proves the corollary 1 too. ■

Note that, in the following, lattice decoding is performed over $\Lambda_{\mathcal{C}}$ (rather than Λ_D). However, we use the term "Construction D" to refer to multilayer convolutional lattices. Considering that lattice codes based on Construction D, as described in the previous subsection, are indeed multilevel convolutional codes [25] and so one can apply multi-stage trellis decoding algorithms for ML, MAP or lattice decoding (similar to Construction A in Section IV-B). The multi-stage decoding is started by decoding the inner code with the largest minimum distance $\mathcal{C}_a$; the decoded layer is then fed to the higher layer $\mathcal{C}_{a-1}$ and is used as *a priori* information for decoding the data of the corresponding layer. The process is continued to until decoding $\mathcal{C}_1$. Note that any known decoding algorithms, e.g. Viterbi or BCJR, can be applied for decoding the layers (we are interested in BCJR decoding in this paper as it is a SISO decoder). The BCJR decoder for multi-stage decoding is slightly different than the Conventional BCJR decoder because the state transition probabilities depend on the *a priori* information of the other layers too. *A priori* information can be hard information plugged from inner layers to upper layers or soft information that only passes the probability of the data corresponding to the other layers; by exchanging soft information among the layers, iterative decoding of the layers is also possible that indeed offers higher performance gains.

Example 2: A design methodology of a two layer lattice based on construction D from $(7, 5)$ convolutional codes is explained in this example. Assume that the nested codes are defined as $\mathbb{F}_2^N \supseteq \mathcal{C}_1 \supseteq \mathcal{C}_2$ where the dimension $N = 300$, $\mathcal{C}_1$ and $\mathcal{C}_2$ together participate in encoding 150 bits (i.e, number of messages $M = 2^{150}$). Moreover, assume $\mathcal{C}_2$ participates in encoding 10 bits and $\mathcal{C}_1$ encodes 140 bits. In the following we will describe obtaining the generator matrix of the lattice based on Construction D where the minimum Euclidean distance

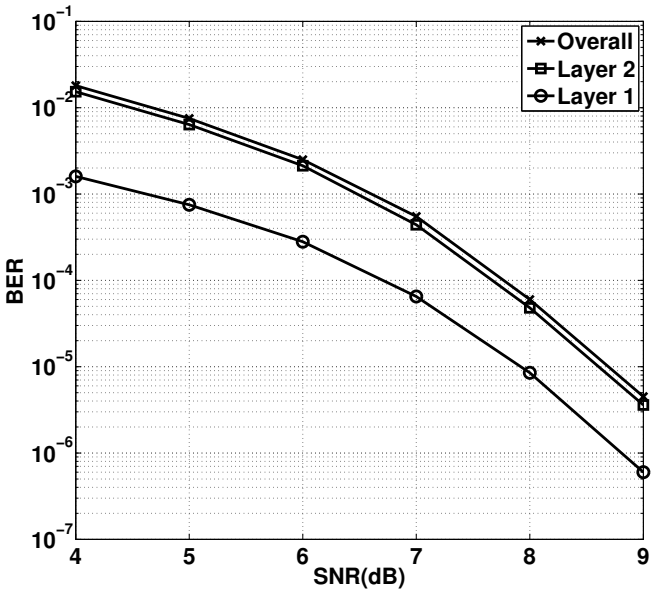

Fig. 10. BER – Construction D

criterion of the Construction D definition is fulfilled. Later on, a corresponding lattice encoder based on conventional convolutional encoders is described.

Lattice Generator Matrix

One lattice based on construction D will be obtained as follows:

- *Inner layer:* the inner code $\mathcal{C}_2$ carries only 10 data bits in inner layer of a codeword of length 300 that is generated from a data word of length 150 (because it is based on a $(7,5)$ convolutional code); consequently, each data bit in inner layer can be represented by 15 "virtual" data bits that can be arranged in the desired arrangement in order to achieve the desired MED. As there are 10 data bits to be encoded in the inner layer, and there are only 10 basis vectors that need to be specified; we propose $\boldsymbol{b}_1 = \oplus r_i$ where $i \in \{1,4,7,10,13\}$ and r_i specifies the rows of the generator matrix of $(7,5)$ code that was derived in (11); $\oplus$ represents mod-2 summation. In other words, $\boldsymbol{b}_1$ is a codeword generated by a data word, in octal notation $\boldsymbol{d}_{\boldsymbol{b}_1} = [4,4,4,4,4,\mathbf{0}_{1\times 135}]$. Hence, feeding $\boldsymbol{d}_{\boldsymbol{b}_1}$ to a $(7,5)$ convolutional encoder generates the first basis vector.
  Likewise, $\boldsymbol{b}_2 = \oplus r_i$ where $i \in \{16,19,22,25,28\}$ that is generated from a data word in octal notation as $\boldsymbol{d}_{\boldsymbol{b}_2} = [\mathbf{0}_{1\times 15},4,4,4,4,4,\mathbf{0}_{1\times 120}]$. One can similarly obtain all the basis vectors of the inner generator matrix; e.g., the 10-th basis vector is $\boldsymbol{b}_{10} = \oplus r_i$ with $i \in \{136,139,142,145,148\}$ that is generated from the $\boldsymbol{d}_{\boldsymbol{b}_{10}} = [\mathbf{0}_{1\times 135},4,4,4,4,4]$ data vector. Consequently, plugging in $\boldsymbol{d}_{\boldsymbol{b}_10}$ to a $(7,5)$ convolutional encoder will generate the basis vectors corresponding to the inner layer, i.e., the first ten rows of Λ_D.
- *Outer layer:*One can arbitrarily choose the 140 remaining basis vectors (i.e.,$\boldsymbol{b}_{11},\boldsymbol{b}_{12},\cdots,\boldsymbol{b}_{150}$) from the generator matrix (11) of the convolutional code; however, the only constraint is that for any $i \geq 11$, the summation $\oplus\ \boldsymbol{b}_i \neq \boldsymbol{b}_k$ where $k = 1,2,\cdots,10$. The constraint is stressed to make sure that the generator matrix of the lattice is full rank. The remaining 150 basis vectors are chosen from $2\boldsymbol{I}_{300\times 300}$ matrix with which G_{Λ_D} is a full rank matrix.

Construction D using CCF: The equivalent shift register based encoder consists of two conventional convolutional encoders where 140 data bits are encoded by $\mathcal{C}_1$ and 10 data bits by $\mathcal{C}_2$; moreover, the data of $\mathcal{C}_2$ are repeated in the corresponding positions to generate a "virtual" data word of length 150. Fig. 10 illustrates BER of the convolutional lattice in *Example 2*: Overall BER is shown by black solid line marked with (x). The BER of the outer layer is nearly equal to the overall BER because the performance of the code is bounded by the MED of the outer layer (note that minimum Euclidean distance of the outer layer is smaller than minimum Euclidean distance of the inner layer). Clearly, due to the large minimum Euclidean distance of inner layer, its BER is much lower.

It was claimed earlier (without proof) that neglecting MED criterion of the definition of Construction D will degrade the error performance of the overall system; in order to validate this, we have provided another simulation in Fig. 11 where $\mathcal{C}_2$ is a $[10,300,12]$. The minimum Euclidean distance is 12 ($d_{\min}^{\mathcal{C}_2} = 12$) which is smaller than the minimum Euclidean distance of the $\mathcal{C}_2$ in the above example. As expected, the BER of both layers is degraded when compared with Fig. 10.

VII. Conclusion and Future Work

Constructing convolutional lattices based on Construction A/D is proposed in this paper. Also, lattice decoding using trellis structure of the underlying convolutional code is discussed. Unlike the existing lattice decoding algorithm the proposed method is practically feasible with reasonable complexity at arbitrarily high dimensions. Moreover, the performance of the proposed lattice decoder is found to be superior, since decoding is performed at lower dimension compared to the dimension of Construction A/D. Furthermore, the statistical characteristics of MLAN are derived in this paper, and are exploited by the BCJR decoder.

Appendix A
Distribution of N'

Assume T is a normally distributed random variable with mean equal to $\mu = \frac{a+b}{2}$ and variance equal to $\alpha^2\sigma^2$, i.e., $f_T(t) = \frac{1}{\sqrt{2\pi}\alpha\sigma}\mathrm{e}^{-\frac{(t-\mu)^2}{2\alpha^2\sigma^2}}$. Also, assume V is a random variable according to a uniform distribution in the (η_b,η_a) interval, i.e.,

$$f_V(v) = \begin{cases} \frac{1}{\eta_a-\eta_b} & \eta_b < v < \eta_a \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

and so, the pdf of $W = T + V$ is

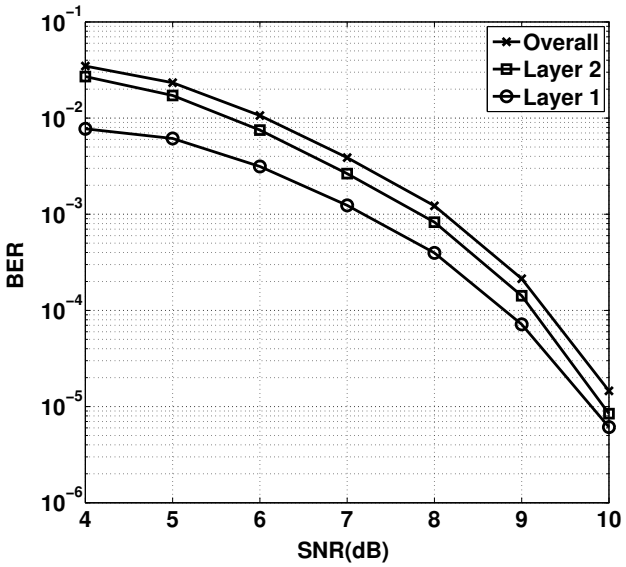

Fig. 11. BER – Construction D

$$f_W(w) = \frac{1}{2(\eta_a-\eta_b)} \times \mathrm{erf}\left(\frac{2w-b-a(3-2\alpha)}{2\sqrt{2}\alpha\sigma}, \frac{2w-a-b(3-2\alpha)}{2\sqrt{2}\alpha\sigma}\right) \quad (35)$$

with $\eta_i = i(1-\alpha)$.

Proof: Assuming $W = T + V$, the pdf of W is the convolution of the pdf of T and V, i.e., $f_W(w) = f_T(t) * f_V(v)$. By resorting to the definition of the convolution operator, one can write

$$f_W(w) = \int_{-\infty}^{\infty} f_V(t) f_T(w-t)\mathrm{d}t = \frac{1}{2(\eta_a-\eta_b)} \int_{\eta_b}^{\eta_a} \frac{1}{\sqrt{2\pi}\alpha\sigma} \mathrm{e}^{-\frac{(w-\mu-t)^2}{2\alpha^2\sigma^2}} \mathrm{d}t. \quad (36)$$

Considering that $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-t^2}\mathrm{d}t$ and $\mu = \frac{a+b}{2}$, after some manipulation, (36) can easily be simplified according to

$$f_W(w) = \frac{1}{2(\eta_a-\eta_b)} \times \mathrm{erf}\left(\frac{2w-b-a(3-2\alpha)}{2\sqrt{2}\alpha\sigma}, \frac{2w-a-b(3-2\alpha)}{2\sqrt{2}\alpha\sigma}\right) \quad (37)$$

■

REFERENCES

[1] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*. Springer, third ed., 1999.

[2] Y. Huang and K. R. Narayanan, "Construction π_a and π_d lattices: Construction, goodness, and decoding algorithms," *CoRR*, vol. abs/1506.08269, 2015.

[3] A. Sakzad, M. Sadeghi, and D. Panario, "Turbo lattices: Construction and error decoding performance," *arXiv:1108.1873*, 2012.

[4] A. Sakzad, M.-R. Sadeghi, and D. Panario, "Construction of turbo lattices," in *48th Annual Allerton Conference on Communication, Control, and Computing*, pp. 14–21, Sept 2010.

[5] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2201–2214, 2002.

[6] A. Ghasemmehdi and E. Agrell, "Faster recursions in sphere decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 6, pp. 3530–3536, 2011.

[7] E. Viterbo and E. Biglieri, "Computing the voronoi cell of a lattice: the diamond-cutting algorithm," *IEEE Transactions on Information Theory*, vol. 42, pp. 161–171, Jan 1996.

[8] M. Eslami and W. A. Krzymien, "Spc05-2: Turbo coded mc-cdm communications with spatial multiplexing," in *IEEE Globecom 2006*, pp. 1–5, IEEE, 2006.

[9] O. Shalvi, N. Sommer, and M. Feder, "Signal codes: Convolutional lattice codes," *IEEE Transactions on Information Theory*, vol. 57, pp. 5203–5226, Aug 2011.

[10] M. M. Molu, "Reduced complexity BCJR algorithms for Faster Than Nyquist (FTN) signalling," *M.Sc. Thesis, Lund University*, 2009.

[11] M. M. Molu and A. Burr, "Constructing convolutional lattices and its application in compute and forward," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, pp. 2187–2193, June 2015.

[12] G. Forney, M. Trott, and S.-Y. Chung, "Sphere-bound-achieving coset codes and multilevel coset codes," *IEEE Transactions on Information Theory*, vol. 46, pp. 820–850, May 2000.

[13] U. Erez and R. Zamir, "Achieving $\frac{1}{2}\log$ (1+SNR) on the AWGN channel with lattice encoding and decoding," *IEEE Transactions on Information Theory*, vol. 50, pp. 2293–2314, Oct 2004.

[14] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Transactions on Information Theory*, vol. 40, pp. 409–417, Mar 1994.

[15] G. W. W. Charles H. Swannack, Uri Erez, "Geometric relationships between gaussian and modulo-lattice error exponents," in *arXiv:1308.1609*.

[16] T. Liu, P. Moulin, and R. Koetter, "On error exponents of modulo lattice additive noise channels," *IEEE Transactions on Information Theory*, vol. 52, pp. 454–471, Feb 2006.

[17] M. Molu and N. Goertz, "Optimal precoding in the relay and the optimality of largest eigenmode relaying with statistical channel state information," *IEEE Transactions on Wireless Communications*, vol. 13, pp. 2113–2123, April 2014.

[18] M. M. Molu, K. Cumanan, and A. Burr, "Low-complexity compute-and-forward techniques for multisource multirelay networks," *IEEE Communications Letters*, vol. 20, pp. 926–929, May 2016.

[19] B. Nazer and M. Gastpar, "Compute-and-Forward: Harnessing interference through structured codes," *IEEE Transactions on Information Theory*, vol. 57, pp. 6463–6486, Oct. 2011.

[20] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm i. expected complexity," *IEEE Transactions on Signal Processing*, vol. 53, pp. 2806–2818, Aug 2005.

[21] H. Vikalo and B. Hassibi, "On the sphere-decoding algorithm ii. generalizations, second-order statistics, and applications to communications," *IEEE Transactions on Signal Processing*, vol. 53, pp. 2819–2834, Aug 2005.

[22] J. Forney, G.D., "A bounded-distance decoding algorithm for the Leech lattice, with generalizations," *IEEE Transactions on Information Theory*, vol. 35, pp. 906–909, Jul 1989.

[23] M.-R. Sadeghi, A. Banihashemi, and D. Panario, "Low-density parity-check lattices: Construction and decoding analysis," *IEEE Transactions on Information Theory*, vol. 52, pp. 4481–4495, Oct 2006.

[24] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Transactions on Information Theory*, vol. 54, pp. 1561–1585, April 2008.

[25] R. H. Morelos-Zaragoza and H. Imai, "Binary multilevel convolutional codes with unequal error protection capabilities," *IEEE Transactions on Communications*, vol. 46, no. 7, pp. 850–853, 1998.

[26] J. Harshan, E. Viterbo, and J.-C. Belfiore, "Construction of Barnes-Wall lattices from linear codes over rings," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 3110–3114, July 2012.

[27] J. Harshan, E. Viterbo, and J.-C. Belfiore, "Practical encoders and decoders for euclidean codes from Barnes-Wall lattices," *IEEE Transactions on Communications*, vol. 61, pp. 4417–4427, November 2013.

[28] Y. Yan, C. Ling, and X. Wu, "Polar lattices: Where Arikan meets Forney," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 1292–1296, July 2013.

[29] W. Kositwattanarerk and F. Oggier, "Connections between Construction D and related constructions of lattices," *arXiv:1308.6175*, 2014.